[0261] In a preferred embodiment, the user can start a slide and specify its chord in either of two ways. In the first way, the user starts with the hand floating above the surface, places some fingers on the surface possibly asynchronously, and begins moving all of these fingers laterally. Decision diamond **656** initiates the slide mode only when significant motion is detected in all the touching fingers. Step **658** selects the chord from the combination of fingers touching when significant motion is detected, regardless of touchdown synchronization. In this case coherent initiation of motion in all the touching fingers is sufficient to distinguish the slide from resting fingers, so synchronization of touchdown is not necessary. Also, novice users may erroneously try to start a slide by placing and sliding only one finger on the surface, forgetting that multiple fingers are necessary. Tolerance of asynchronous touchdown allows them to seamlessly correct this by subsequently placing and sliding the rest of the fingers desired for the chord. The slide chord will then initiate without forcing the user to pick up all fingers and start over with synchronized finger touchdowns.

[0262] In the second way, the user starts with multiple fingers resting on the surface, lifts a subset of these fingers, touches a subset back down on the surface synchronously to select the chord, and begins moving the subset laterally to initiate the slide. Decision diamond **656** actually initiates the slide mode when it detects significant motion in all the fingers of the synchronized subset. Whether the fingers which remained resting on the surface during this sequence begin to move does not matter since in this case the selected chord is determined in step **658** by the combination of fingers in the synchronized press subset, not from the set of all touching fingers. This second way has the advantage that the user does not have to lift the whole hand from the surface before starting the slide, but can instead leave most of the weight of the hands resting on the surface and only lift and press the two or three fingers necessary to identify the most common finger chords.

[0263] To provide greater tolerance for accidental shifts in resting finger positions, decision diamond **656** requires both that all relevant fingers are moving at significant speed and that they are moving about the same speed. This is checked either by thresholding the geometric mean of the finger speeds or by thresholding the fastest finger's speed and verifying that the slowest finger's speed is at least a minimum fraction of the fastest finger's speed. Once a chord slide is initiated, step **660** disables recognition of key or chord taps by the hand at least until either the touching fingers or the synced subset lifts off.

[0264] Once the slide initiates, the chord motion recognizer could simply begin sending raw component velocities paired with the selected combination of finger identities to the host. However, in the interest of backward compatibility with the mouse and key event formats of conventional input devices, the motion event generation steps in FIG. **40**B convert motion in any of the extracted degrees of freedom into standard mouse and key command events which depend on the identity of the selected chord. To support such motion conversion, step **658** finds a chord activity structure in a lookup table using a bitfield of the identities of either the touching fingers or the fingers in the synchronized, subset. Different finger identity combinations can refer to the same chord activity structure. In the preferred embodiment, all finger combinations with the same number of non-thumb fingertips refer to the same chord activity structure, so slide chord activities are distinguished by whether the thumb is touching and how many non-thumb fingers are touching. Basing chord action on the number of fingertips rather than their combination still provides up to seven chords per hand yet makes chords easier for the user to memorize and perform. The user has the freedom to choose and vary which fingertips are used in chords requiring only one; two or three fingertips. Given this freedom, users naturally tend to pick combinations in which all touching fingertips are adjacent rather than combinations in which a finger such as the ring finger is lifted but the surrounding fingers such as the middle and pinky must touch. One chord typing study found that users can tap these finger chords in which all pressed fingertips are adjacent twice as fast as other chords.

[0265] The events in each chord activity structure are organized into slices. Each slice contains events to be generated in response to motion in a particular range of speeds and directions within the extracted degrees of freedom. For example, a mouse cursor slice could be allocated any translational speed and direction. However, text cursor manipulation requires four slices, one for each arrow key, and each arrow's slice integrates motion in a narrow direction range of translation. Each slice can also include motion sensitivity and so-called cursor acceleration parameters for each degree of freedom. These will be used to discretize motion into the units such as arrow key clicks or mouse clicks expected by existing host computer systems.

[0266] Step **675** of chord motion conversion simply picks the first slice in the given chord activity structure for processing. Step **676** scales the current values of the extracted velocity components by the slice's motion sensitivity and acceleration parameters. Step **677** geometrically projects or clips the scaled velocity components into the slice's defined speed and direction range. For the example mouse cursor slice, this might only involve clipping the rotation and scaling components to zero. But for an arrow key slice, the translation velocity vector is projected onto the unit vector pointing in the same direction as the arrow. Step **678** integrates each scaled and projected component velocity over time in the slice's accumulators until decision diamond **680** determines at least one unit of motion has been accumulated. Step **682** looks up the slice's preferred mouse, key, or three-dimensional input event format, attaches the number of accumulated motion units to the event; and step **684** dispatches the event to the outgoing queue of the host communication interface **20**. Step **686** subtracts the sent motion events from the accumulators, and step **688** optionally clears the accumulators of other slices. If the slice is intended to generate a single key command per hand motion, decision diamond **689** will determine that it is a one-shot slice so that step **690** can disable further event generation from it until a slice with a different direction intervenes. If the given slice is the last slice, decision diamond **692** returns to step **650** to await the next scan of the sensor array. Otherwise step **694** continues to integrate and convert the current motion for other slices.

[0267] Returning to FIG. **40**A, for some applications it may be desirable to change the selected chord whenever an additional finger touches down or one of the fingers in the chord lifts off. However, in the preferred embodiment, the selected chord cannot be changed after slide initiation by asynchronous finger touch activity. This gives the user freedom to rest or lift addition fingers as may be necessary to get the best precision in a desired degree of freedom. For example, even though the finger pair chord does not include the thumb, the thumb can be set down shortly after slide initiation to access